Let x = G

Graph Neural Networks

Anton Baykalov

MathLearn — June 2026

Graphs are everywhere in mathematics

"Graphs are everywhere in mathematics but once you start thinking about them, they are even more everywhere."

Geordie Williamson
Sydney Mathematical Research Institute

Graphs and tasks

- Let $G = (V, E)$ denote a graph with vertex feature vectors $x_v \in \mathbb{R}^{k_0}$ for each $v \in V$.

There are two tasks of interest:

- **Node(vertex) clasification**: each node has an associated label z_v and the goal is to learn a representation vector h_v such that the lable can be predicted as $z_v = f(h_v)$.
  - Graph colouring
  - Predict if v is in a 3-cycle
  - $V = \{1, 2, \ldots, 30\}$ where there is an edge between m and n whenever $m \mid n$. Predicts primality.
- **Graph classification**: given a set of graphs and their lables, learn a representation vector that helps predict the label of the entire graph, $z_G = g(h_G)$.
  - Is a graph (k-)regular?
  - Does a graph contain a (k-)cycle?

Permutation invariance and equivariance

Why new archetecture for learning graph representations?

- Our current tool box is not enough
  - CNNs are well defined only over grid-structured inputs
  - RNNs are well defined only over sequences
- Good GNN should be **permutationaly** invariant/equivariant:
  - A straitforward (reasonable) idea is to represent a graph as an **adjacency matrix**, flattent it and feed to **MLP**:

$$z_G = \text{MLP}(A[1] \oplus A[2] \oplus \ldots \oplus A[|V|])$$

  - This depends on the arbitrary ordering of nodes that used in the adjacency matrix
  - We want any function f that takes an adjacency matrix A as input to satisfy one of the two following properties:
    * $f(PAP^\top) = f(A)$ – **permutation invariance**
    * $f(PAP^\top) = P \cdot f(A)$ – **permutational equivariane**

GNNs in one picture

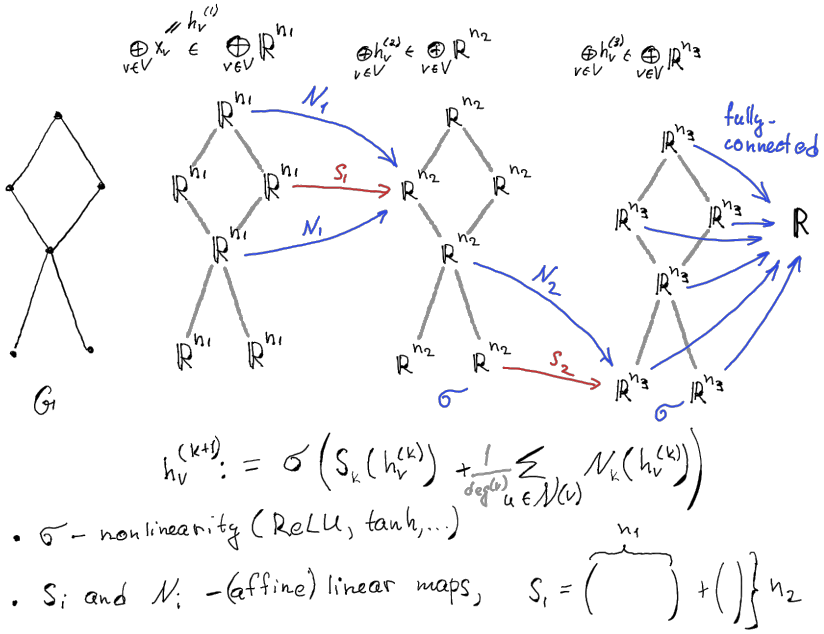

Figure 1: GNN Architecture

Message passing and its variations

- During message-passing itiration of a GNN, a **hidden- embedding** $n_v^{(k)}$ is updated according to information a **aggregetaed** for the neighbourhood $\mathcal{N}(v)$:

$$h_v^{(k+1)} = \sigma\left(S_k(h_v^{(k)}) + \sum_{u \in \mathcal{N}(v)} N_k(h_u^{(k)})\right)$$

$$h_v^{(k+1)} = \text{UPDATE}^{(k)}\left(h_v^{(k)}, \text{AGGREGATE}^{(k)}(h_u^{(k)}, \forall u \in \mathcal{N}(v))\right) \quad (1)$$

$$= \text{UPDATE}^{(k)}\left(h_v^{(k)}, \mathbf{m}_{\mathcal{N}(v)}^{(k)}\right) \quad (2)$$

- **In general,** UPDATE and AGGREGATE are arbitrary differentiable functions (i.e. newral networks).

Message passing and its variations

- Message passing with **self-loop**:

$$h_v^{(k+1)} = \text{AGGREGATE}^{(k)}(h_u^{(k)}, \forall u \in \mathcal{N}(v) \cup \{v\})$$

- Neighborhood **normalisation**:

$$\mathbf{m}_{\mathcal{N}(v)} = \frac{\sum_{u \in \mathcal{N}(v)} N_k(h_u^{(k)})}{deg(v)}$$

- **Janossy Pooling**:

$$\mathbf{m}_{\mathcal{N}(v)} = \text{MLP}_\theta\left(\frac{1}{|\Pi|}\sum_{\pi \in \Pi} \rho_\phi(h_{u_1}, h_{u_2}, \ldots, h_{u_{deg_v}})_\pi\right)$$

- **Attention** – assign attention weight to each neighbour

How powerful are GNNs? (theoretically)

- **Weisfeiler-Lehman test** for graph isomorphism.
  - WL test of graph isomorphism (Weisfeiler & Lehman, 1968) is an effective andcomputationally efficient test that distinguishes a broad class of graphs (Babai & Kucera, 1979).

- Its 1-dimensional form, "naïve vertex refinement", is analogous to neighbor aggregation in GNNs: WL test iteratively

(1) aggregates the labels of nodes and their neighborhoods,
(2) hashes the aggregated labels into unique new labels.

- The algorithm decides that two graphs are non-isomorphic if at some iteration the labels of the nodes between the two graphs differ.

Lemma. *Let G_1 and G_2 be any two non-isomorphic graphs. If a GNN $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ maps G_1 and G_2 to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides G_1 and G_2 are not isomorphic.*

How powerful are GNNs?

Theorem. *Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN. With a sufficient number of GNN layers, $\mathcal{A}$ maps any graphs G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:*

1. *$\mathcal{A}$ aggregates and updates node features iteratively with $h_v^{(k+1)} = \phi\left(h_v^{(k)}, f(\{h_u^{(k)} \mid u \in \mathcal{N}(v)\})\right)$ where the functions f, which operates on multisets, and ϕ are injective.*
2. *$\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\{h_v^{(k)}\}$ is injective.*

What GNNs are good for on practice?

Local structure: GNNs are very good at tasks that are determined by a node's immediate neighbourhood - things a k-hop walk can see.

- Degree and local connectivity (is this node a hub? a leaf? a cut vertex?)
- Local clustering (does this node sit inside a dense clique or on the boundary between communities)
- Short cycle membership (k-layer GNN can detect cycles of length up to $2k + 1$ in principle)

Global structure: GNNs are not naturally good at tasks that require global reasoning, because aggregation is local by design.

- Diameter, shortest paths, graph radius (require seeing the whole graph, not just neighbourhoods)
- Maximum independent set, graph colouring (require symmetry-breaking that local aggregation cannot achieve
- Hamiltonicity, graph genus (fundamentally global, NP-hard, GNNs plateau quickly)